US006457101B1

(12) **United States Patent**
Bauman et al.

(10) **Patent No.: US 6,457,101 B1**
(45) **Date of Patent: Sep. 24, 2002**

(54) **SYSTEM AND METHOD FOR PROVIDING THE SPECULATIVE RETURN OF CACHED DATA WITHIN A HIERARCHICAL MEMORY SYSTEM**

(75) Inventors: **Mitchell A. Bauman**, Circle Pines, MN (US); **Roger L. Gilbertson**, Minneapolis, MN (US); **Donald R. Kalvestrand**, Lonsdale, PA (US); **Joseph S. Schibinger**, Phoenixville, PA (US); **Daniel S. Tokoly**, Huntingdon Valley, PA (US)

(73) Assignee: **Unisys Corporation**, Blue Bell, PA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/468,050**

(22) Filed: **Dec. 20, 1999**

(51) **Int. Cl.**[7] **G06F 12/08**
(52) **U.S. Cl.** **711/122**; 711/143; 711/145
(58) **Field of Search** 711/122, 143, 711/145

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,755,930	A *	7/1988	Wilson et al.	711/122
4,843,542	A *	6/1989	Dashiell et al.	711/119
5,025,365	A *	6/1991	Mathur et al.	711/121
5,423,016	A *	6/1995	Tsuchiya et al.	711/123
6,321,296	B1 *	11/2001	Pescatore	711/117

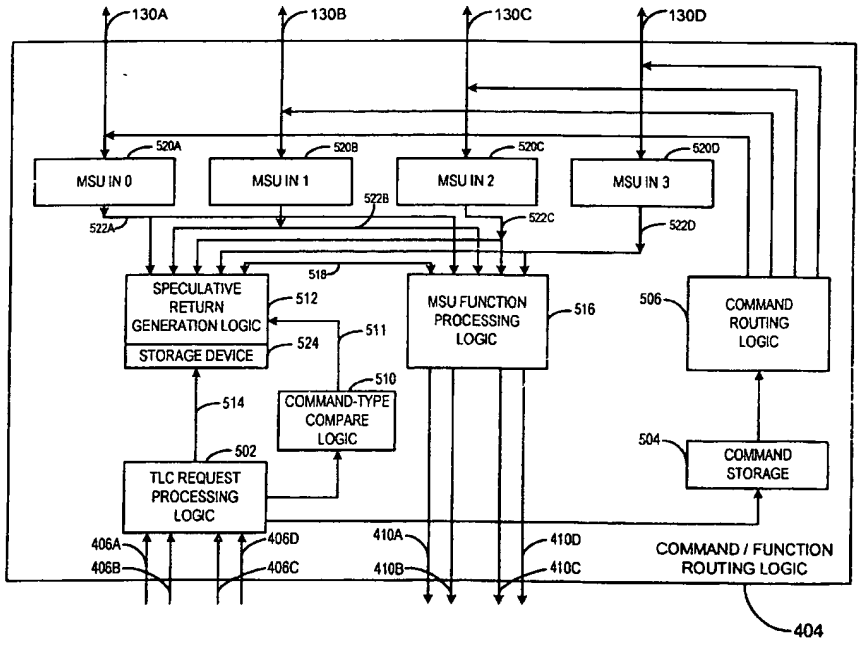
* cited by examiner

Primary Examiner—Reginald G. Bragdon
(74) *Attorney, Agent, or Firm*—Beth L. McMahon; Charles A. Johnson; Mark T. Starr

(57) **ABSTRACT**

A hierarchical memory structure includes a directory-based main memory coupled to multiple first storage devices, each to store data signals retrieved from the main memory. Ones of the first storage devices are further respectively coupled to second storage devices, each to store data signals retrieved from the respectively coupled first storage devices. Fetch requests to retrieve data signals are issued by ones of the storage devices to the main memory. In response, the main memory determines where the most recent data copy resides, and issues a return request, if necessary to retrieve that copy for the requesting storage device. A speculative return generation logic circuit is coupled to at least two of the first storage devices to intercept the fetch requests. In response to an intercepted request, the speculative return generation logic circuit generates a speculative return request directly to one or more of the other coupled first storage devices. This speculative return request causes any updated copies of the requested data signals that may be stored at a lower level in the hierarchical memory, to be transferred to the first storage device. If a return request for the data is then issued by the main memory in response to the fetch request, the requested data signals are resident in a first storage device, and are readily available to the main memory.

20 Claims, 8 Drawing Sheets



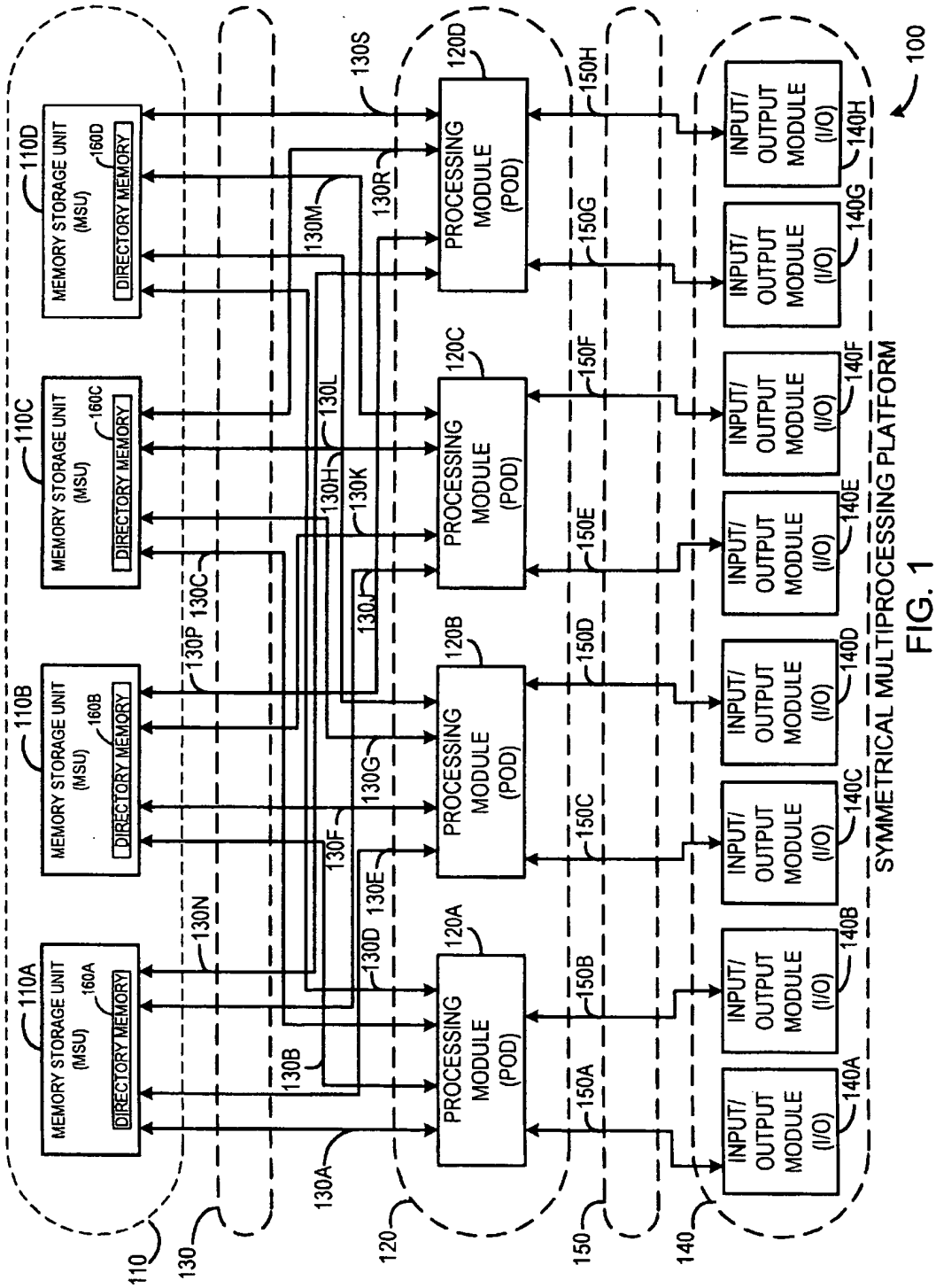
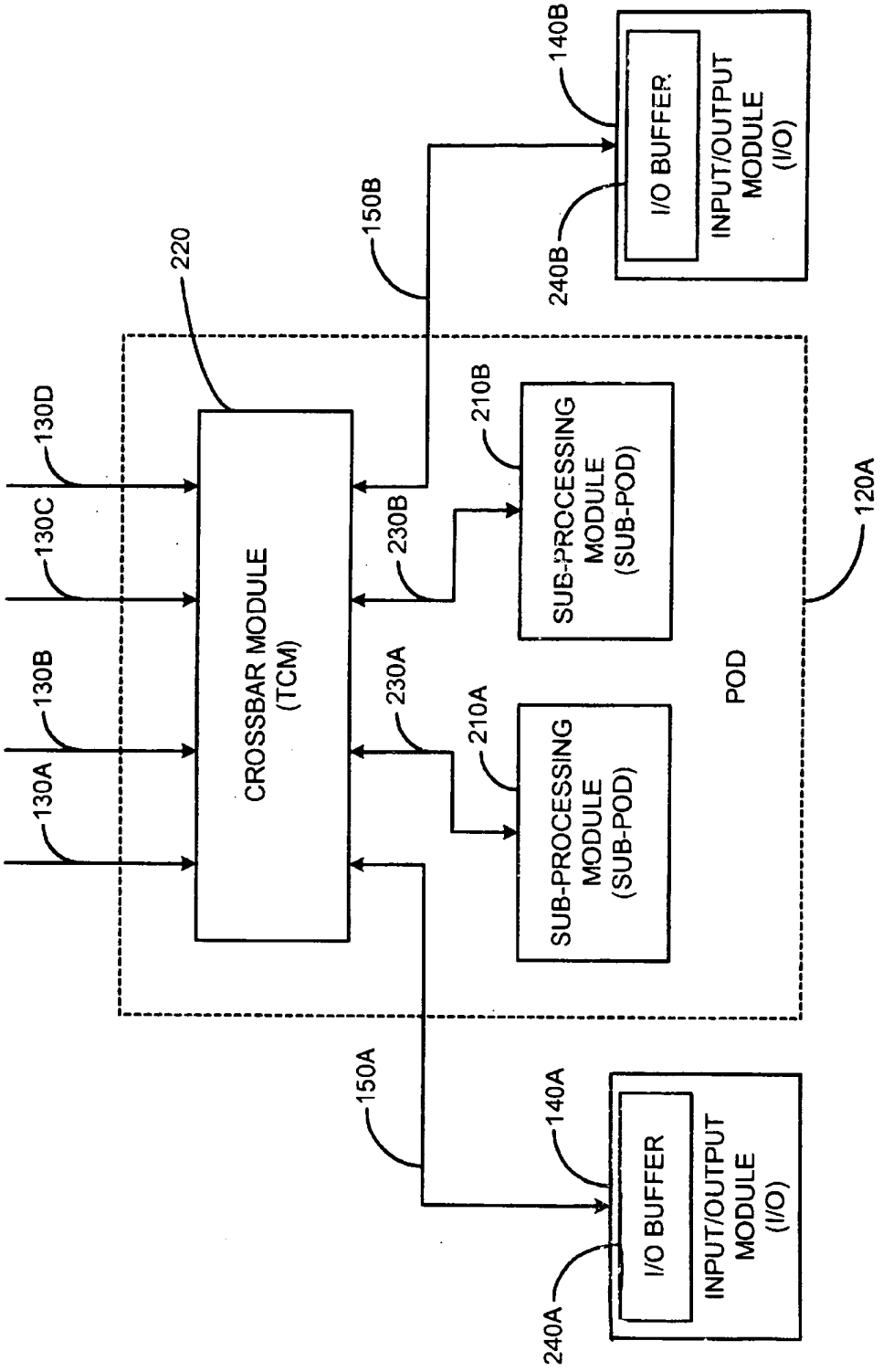
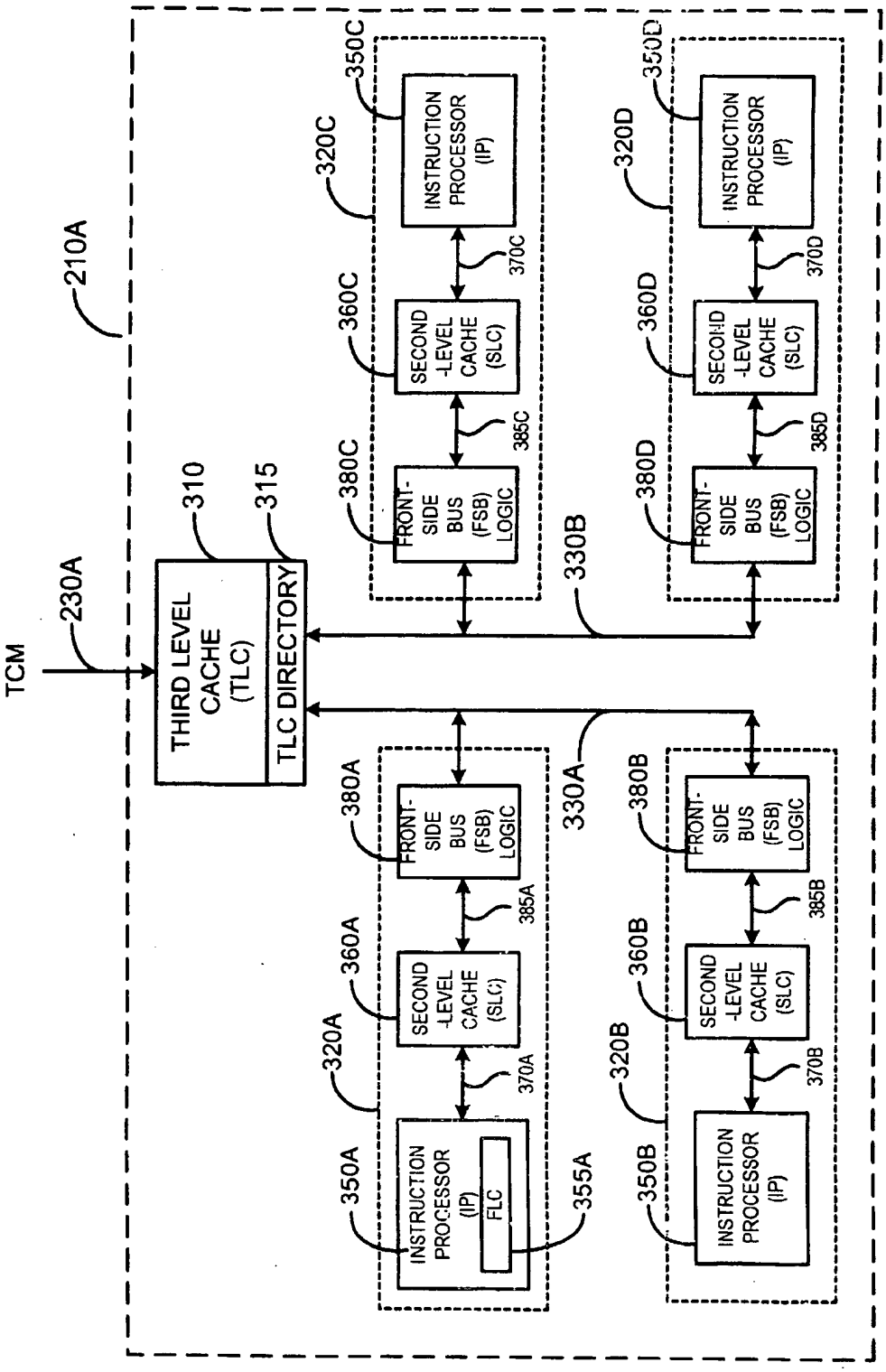

100
SYMMETRICAL MULTIPROCESSING PLATFORM
110
130
120
150
140
MEMORY STORAGE UNIT (MSU)
DIRECTORY MEMORY
110A
160A
110B
160B
110C
160C
110D
160D
PROCESSING MODULE (POD)
120A
120B
120C
120D
INPUT/ OUTPUT MODULE (I/O)
140A
140B
140C
140D
140E
140F
140G
140H
130A
130B
130C
130D
130E
130F
130G
130H
130J
130K
130L
130M
130N
130P
130R
130S
150A
150B
150C
150D
150E
150F
150G
150H

FIG. 1



PROCESSING MODULE (POD)
FIG. 2



SUB-PROCESSING MODULE (SUB-POD)

FIG. 3

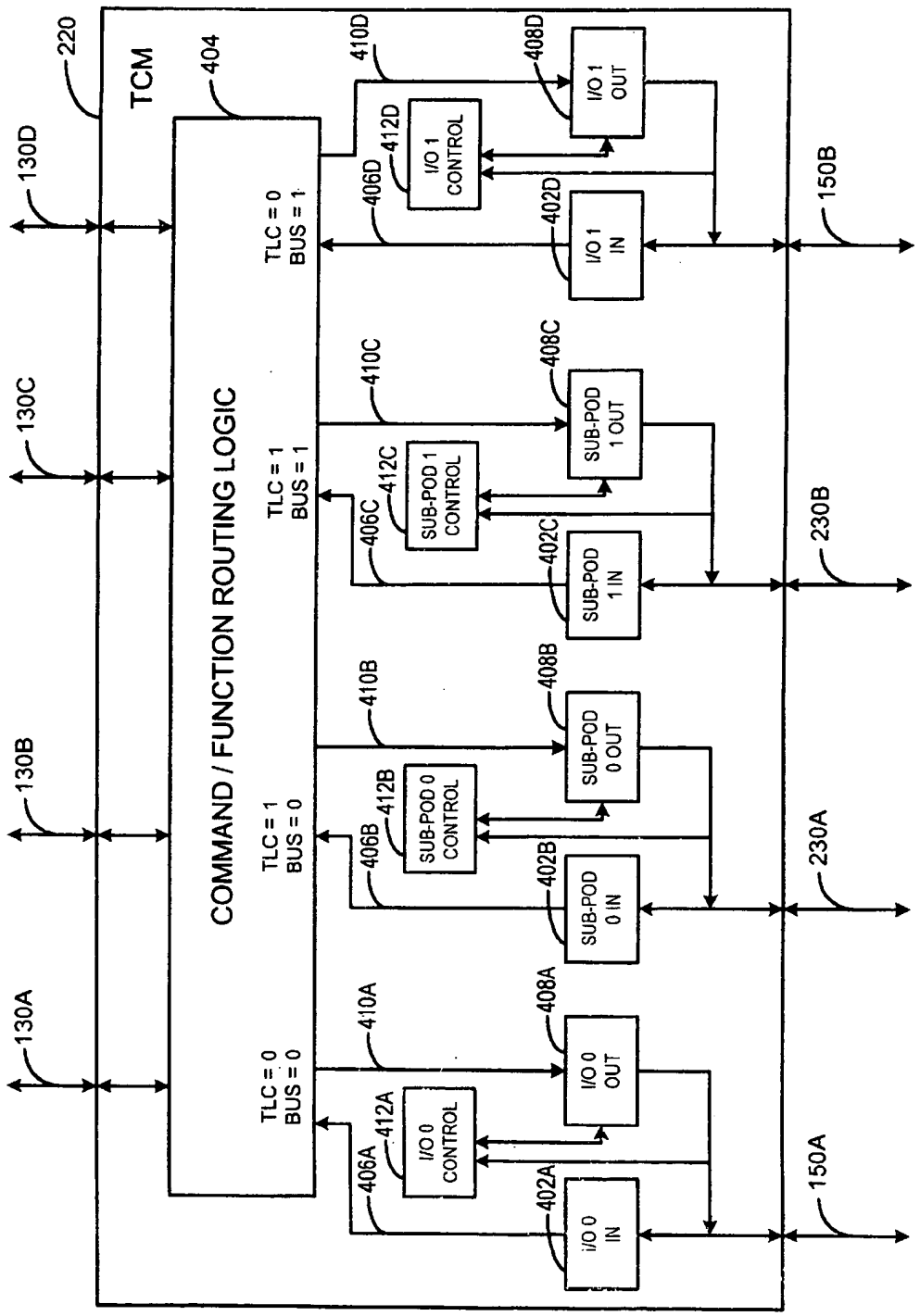

220
TCM
404
COMMAND / FUNCTION ROUTING LOGIC
130D
130C
130B
130A
150B
230B
230A
150A
TLC = 0
BUS = 1
TLC = 1
BUS = 1
TLC = 1
BUS = 0
TLC = 0
BUS = 0
410D
408D
I/O 1 OUT
412D
I/O 1 CONTROL
406D
402D
I/O 1 IN
410C
408C
SUB-POD 1 OUT
412C
SUB-POD 1 CONTROL
406C
402C
SUB-POD 1 IN
410B
408B
SUB-POD 0 OUT
412B
SUB-POD 0 CONTROL
406B
402B
SUB-POD 0 IN
410A
408A
I/O 0 OUT
412A
I/O 0 CONTROL
406A
402A
I/O 0 IN

FIG. 4

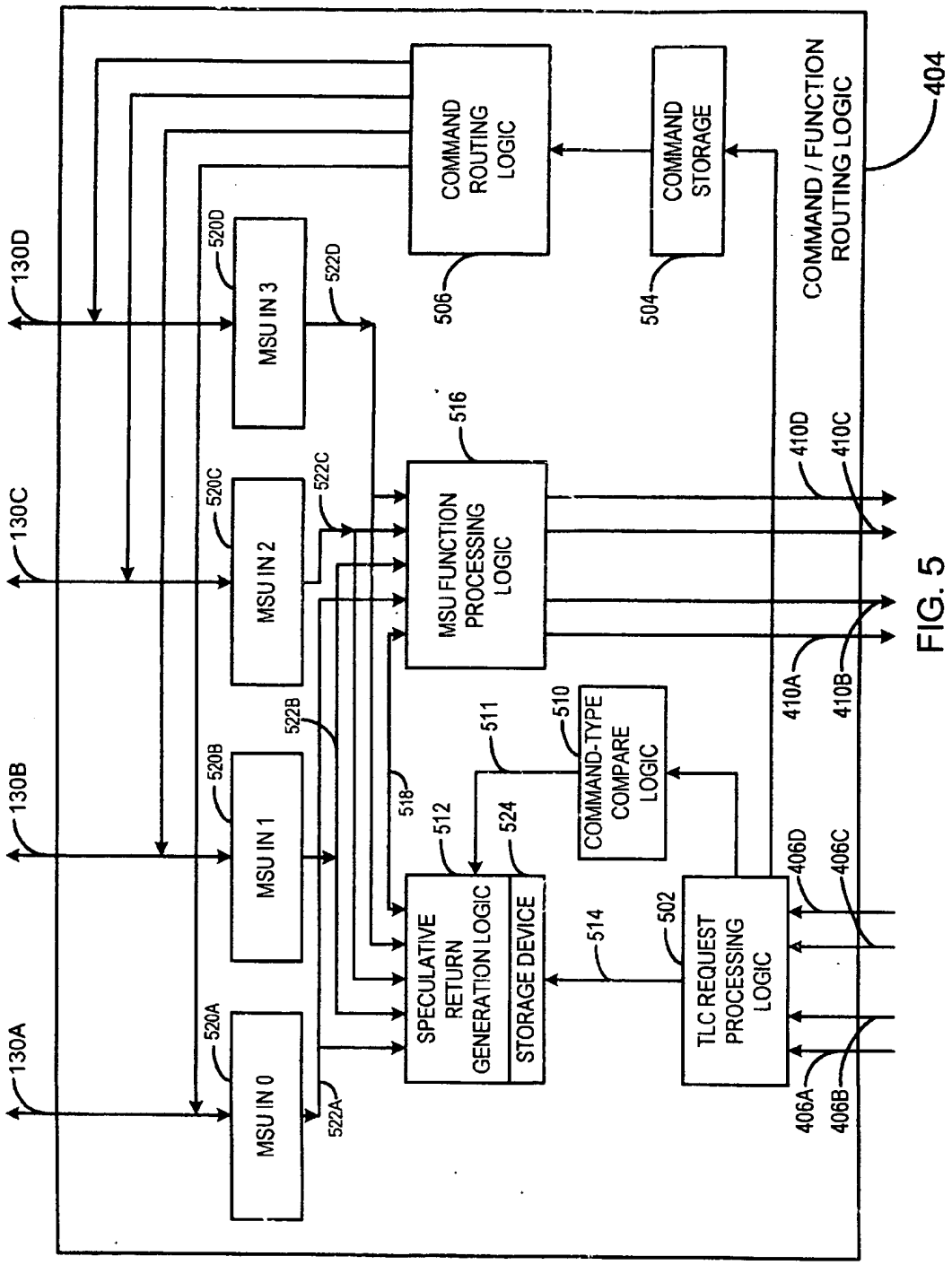

COMMAND / FUNCTION ROUTING LOGIC
404
COMMAND ROUTING LOGIC
506
COMMAND STORAGE
504
MSU IN 3
MSU IN 2
MSU IN 1
MSU IN 0
130D
130C
130B
130A
520D
520C
520B
520A
522D
522C
522B
522A
MSU FUNCTION PROCESSING LOGIC
516
SPECULATIVE RETURN GENERATION LOGIC
STORAGE DEVICE
512
524
COMMAND-TYPE COMPARE LOGIC
510
511
518
TLC REQUEST PROCESSING LOGIC
502
514
410A
410B
410C
410D
406A
406B
406C
406D

FIG. 5

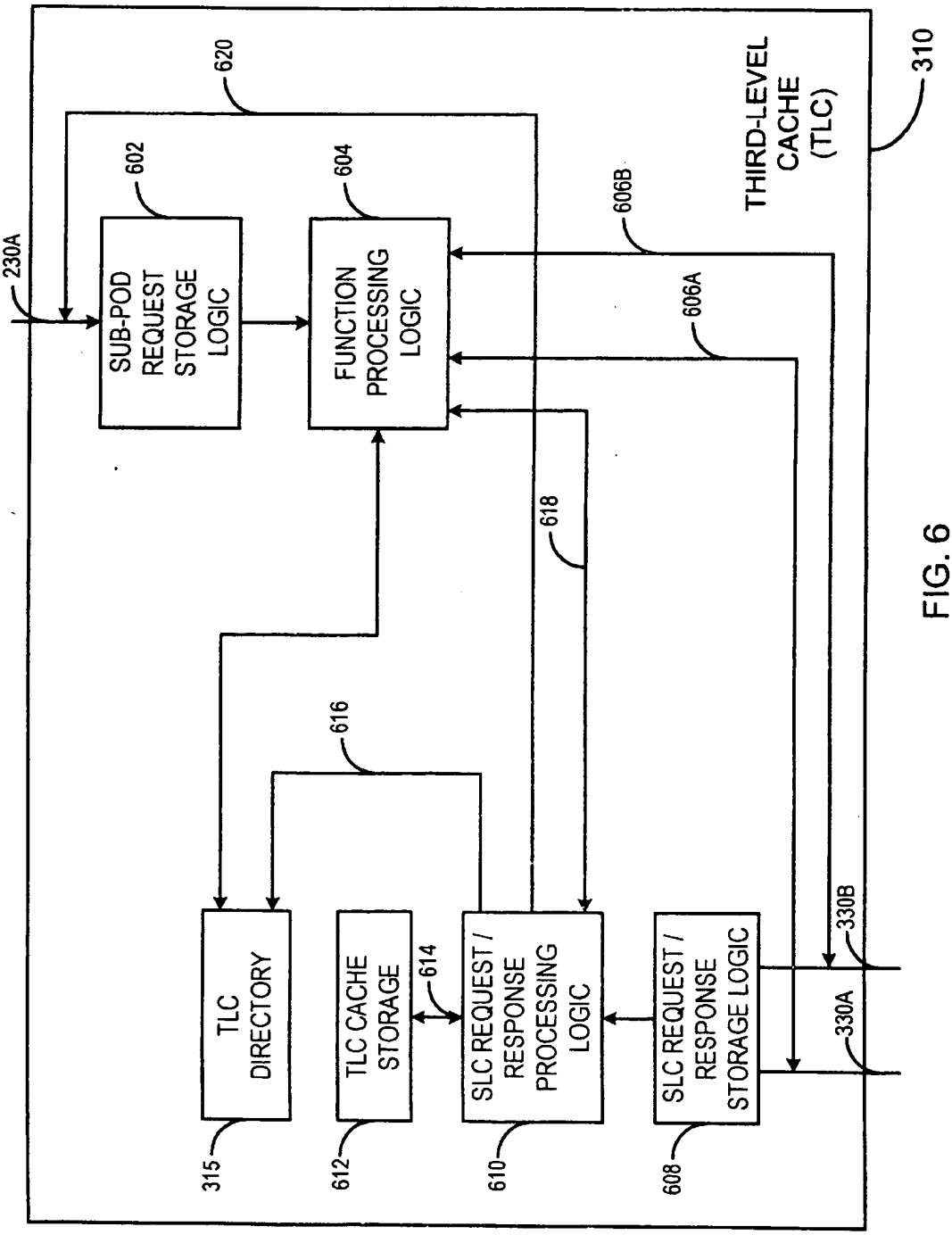

THIRD-LEVEL CACHE (TLC)
310
620
602
604
606B
606A
230A
SUB-POD REQUEST STORAGE LOGIC
FUNCTION PROCESSING LOGIC
618
616
TLC DIRECTORY
TLC CACHE STORAGE
614
SLC REQUEST / RESPONSE PROCESSING LOGIC
SLC REQUEST / RESPONSE STORAGE LOGIC
330B
330A
315
612
610
608

FIG. 6

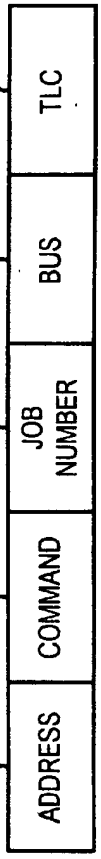


FIG. 7

FIG. 8

902 FETCH COMMAND	904 SPECULATIVE RETURN FUNCTION	906 TLC CACHE STATUS	908 BUS PROBE
FETCH COPY	NONE	N/A	N/A
FETCH ORIGINAL	RETURN ORIGINAL	EXCLUSIVE	EXCLUSIVE BUS PROBES
FETCH CONDITIONAL	RETURN COPY	EXCLUSIVE	SHARED BUS PROBES

FIG. 9

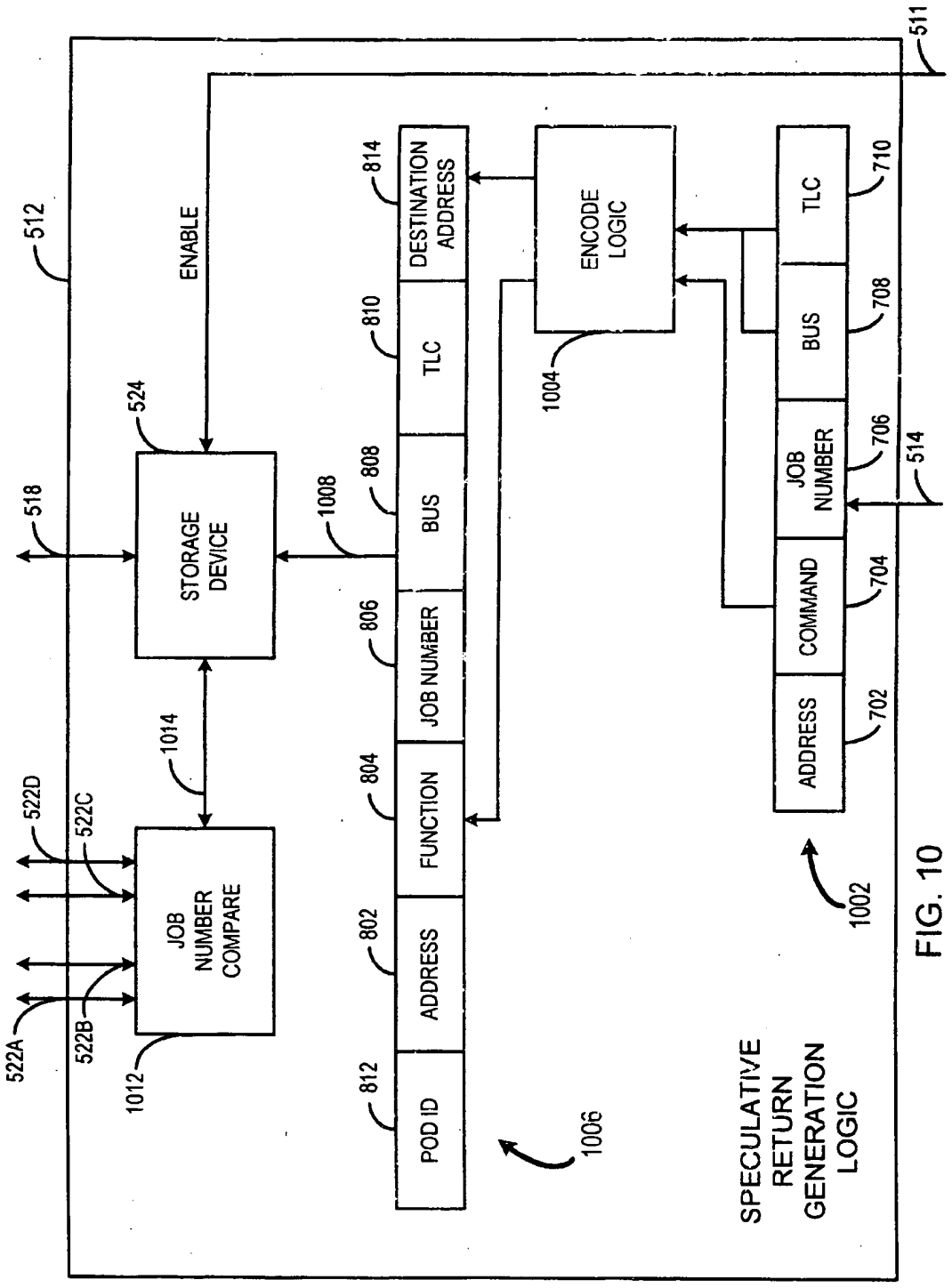

511
512
ENABLE
524
STORAGE DEVICE
518
1008
522D
522C
1014
JOB NUMBER COMPARE
522A
522B
1012
814
DESTINATION ADDRESS
810
TLC
808
BUS
806
JOB NUMBER
804
FUNCTION
802
ADDRESS
812
POD ID
1006
ENCODE LOGIC
1004
710
TLC
708
BUS
706
JOB NUMBER
514
704
COMMAND
702
ADDRESS
1002
SPECULATIVE RETURN GENERATION LOGIC

FIG. 10

SYSTEM AND METHOD FOR PROVIDING THE SPECULATIVE RETURN OF CACHED DATA WITHIN A HIERARCHICAL MEMORY SYSTEM

CROSS-REFERENCE TO OTHER APPLICATIONS

The following co-pending applications of common assignee contain some common disclosure:

"A Directory-Based Cache Coherency System", filed Nov. 05, 1997, Ser. No. 08/965,004, incorporated herein by reference in its entirety;

"High-Speed Memory Storage Unit for a Multiprocessor System Having Integrated Directory and Data Storage Subsystems", filed Dec. 31, 1997, Ser. No. 09/001,588, incorporated herein by reference in its entirety; and

"Directory-Based Cache Coherency System Supporting Multiple Instruction Processor and Input/Output Caches", filed Dec. 31, 1997, Ser. No. 09/001,598, incorporated herein by reference in its entirety.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to an improved hierarchical memory system shared between multiple processors; and more particularly, relates to a memory system that performs transfers of cached data between hierarchical levels of the memory in anticipation of receiving requests to retrieve the data, the transfers being performed so that the data is more readily available to the requester when the anticipated request is received.

2. Description of the Prior Art

Data processing systems are becoming increasing complex. Some systems, such as Symmetric Multi-Processor (SMP) computer systems, couple two or more Instruction Processors (IPs) and multiple Input/Output (I/O) Modules to shared memory. This allows the multiple IPs to operate simultaneously on the same task, and also allows multiple tasks to be performed at the same time to increase system throughput. As the number of units coupled to a shared memory increases, more demands are placed on the memory and memory latency increases. To address this problem, high-speed local memory systems, including caches and high-speed I/O buffer memories, are often coupled to one or more of the IPs for storing data signals that are copied from main memory. These memories are generally capable of processing requests faster than the main memory while also serving to reduce the number of requests that the main memory must handle. This increases system throughput.

While the use of local memories increases system throughput, it causes other design challenges. When multiple local memories are coupled to a single main memory for the purpose of temporarily storing data signals, some system must be utilized to ensure that all IPs and I/O Modules are working from the same (most recent) copy of the data. For example, if a copy of a data item is stored, and subsequently modified, in a cache memory, another IP requesting access to the same data item must be prevented from using the older copy of the data item stored either in main memory or the requesting IP's cache. This is referred to as maintaining cache coherency. Maintaining cache coherency becomes more difficult as more caches are added to the system since more copies of a single data item may have to be tracked.

Many methods exist to maintain cache coherency. Some earlier systems achieve coherency by implementing memory

locks. That is, if an updated copy of data exists within a local cache or buffer memory, other processors are prohibited from obtaining a copy of the data from main memory until the updated copy is returned to main memory, thereby releasing the lock. For complex systems, the additional hardware and/or operating time required for setting and releasing the locks within main memory cannot be justified. Furthermore, reliance on such locks directly prohibits certain types of applications such as parallel processing.

Another method of maintaining cache coherency is shown in U.S. Pat. No. 4,843,542 issued to Dashiell et al., and in U.S. Pat. No. 4,755,930 issued to Wilson, Jr. et al. These patents discuss a system wherein each processor has a local cache coupled to a shared memory through a common memory bus. Each processor is responsible for monitoring, or "snooping", the common bus to maintain currency of its own cache data. These snooping protocols increase processor overhead, and are unworkable in hierarchical memory configurations that do not have a common bus structure. A similar snooping protocol is shown in U.S. Pat. No. 5,025,365 to Mathur et al., which teaches a snooping protocol that seeks to minimize snooping overhead by invalidating data within the local caches at times when other types of cache operations are not occurring. However, the Mathur system can not be implemented in memory systems that do not have a common bus structure.

Another method of maintaining cache coherency is shown in U.S. Pat. No. 5,423,016 to Tsuchiya assigned to the assignee of the current invention. The method described in this patent involves providing a memory structure called a "duplicate tag" that is associated with each cache memory. Each duplicate tag records which data items are stored within the associated cache. When a data item is modified by a processor, an invalidation request is routed to all of the other duplicate tags in the system. The duplicate tags are searched for the address of the referenced data item. If found, the data item is marked as invalid in the other caches. Such an approach is impractical for distributed systems having many caches interconnected in a hierarchical fashion because the time required to route the invalidation requests poses an undue overhead.

For distributed systems having hierarchical memory structures, a directory-based coherency system becomes more practical. Directory-based coherency systems utilize a centralized directory to record the location and the status of data as it exists throughout the system. For example, the directory records which caches have a copy of the data, and further records if any of the caches have an updated copy of the data. When a cache makes a request to main memory for a data item, the central directory is consulted to determine where the most recent copy of that data item resides. Based on this information, the most recent copy of the data is retrieved so that it may be provided to the requesting cache. The central directory is then updated to reflect the new status for that unit of memory. A novel directory-based cache coherency system for use with multiple Instruction Processors coupled to a hierarchical cache structure is described in the co-pending application entitled "Directory-Based Cache Coherency System Supporting Multiple Instruction Processor and Input/Output Caches" referenced above and which is incorporated herein by reference in its entirety.

The use of the afore-mentioned directory-based cache coherency system provides an efficient mechanism for sharing data between multiple processors that are coupled to a distributed, hierarchical memory structure. Using such a system, the memory structure may be incrementally expanded to include any multiple levels of cache memory

while still maintaining the coherency of the shared data. As the number of levels of hierarchy in the memory system is increased, however, some efficiency is lost when data requested by one cache memory in the system must be retrieved from another cache.

As an example of performance degradation associated with memory requests in a hierarchical cache memory system, consider a system having a main memory coupled to three hierarchical levels of cache memory. In the exemplary system, multiple third-level caches are coupled to the main memory, multiple second-level caches are coupled to each third-level cache, and at least one first-level cache is coupled to each second-level cache. This exemplary system includes a non-inclusive caching scheme. This means that all data stored in a first-level cache is not necessarily stored in the interconnected second-level cache, and all data stored in a second-level cache is not necessarily stored in the coupled third-level cache.

Within the above-described system, one or more processors are respectively coupled to make memory requests to an associated first-level cache. Requests for data items not resident in the first-level cache are forwarded on to the intercoupled second-level, and in some cases, the third-level caches. If neither of the intercoupled second or third level caches stores the requested data, the request is forwarded to main memory.

Assume that in the current example, a processor makes a request to the intercoupled first-level cache for a read-only copy of specified data. Assume further that the requested data is not stored in this first-level cache. However, another first-level cache within the system stores a read-only copy of the data. Since the copy of the data is read-only, the request can be completed without involving the other first-level cache. That is, the request may be processed by one of the interconnected second or third-level caches, or if neither of these caches has a copy of the data, by the main memory.

In addition to requests for read-only copies of data, requests may be made to obtain "exclusive" copies of data that can be updated by the requesting processor. In these situations, any previously cached copies of the data must be marked as invalid before the request can be granted to the requesting cache. That is, in these instances, copies of the data may not be shared among multiple caches. This is necessary so that there is only one "most-current" copy of the data existing in the system and no processor is working from outdated data. Returning to the current example, assume the request to the first-level cache is for an exclusive copy of data. This request must be passed via the cache hierarchy to the main memory. The main memory forwards this request back down the hierarchical memory structure to the first-level cache that stores the requested data. This first-level cache must invalidate its stored copy of the data, indicating that this copy may no longer be used. If this first-level cache had an exclusive copy of the data, and had further modified the data, the modified data is passed back to the main memory to be stored in the main memory and to be forwarded on to the requesting first-level cache. In this manner, the requesting cache is provided with an exclusive copy of the most recent data.

The steps outlined above with respect to the exclusive data request are similar to those that must be executed if a read-only copy of the data is requested when a copy of the requested data resides exclusively in another cache. The previous exclusive owner must forward a copy of the updated data to main memory to be returned to the requester.

As may be seen from the current example, in a hierarchical memory system having multiple levels of cache that are not all interconnected by a common bus structure, obtaining an exclusive copy of data that can be utilized by a processor for update purposes may be time-consuming. As the number of these so-called "ownership" requests for obtaining an exclusively "owned" data copy increases within the system, throughput may decrease. This is especially true as additional levels of hierarchy are included in the memory structure.

One mechanism for increasing throughput involves providing a high-speed data return path within the main memory. When data is returned from a previous owner, the high-speed interface forwards the data directly to the requester without the need to perform any type of main memory access. A high-speed interface of this type can be used to route both modified and unmodified data between the various units in the system. Such a system is described in the U.S. patent application entitled "System and Method for By-Passing Supervisory Memory Intervention for Data Transfers Between Devices Having Local Memories", Pat. No. 6,167,489, issued Dec. 26, 2000. While this type of interface decreases the time required to complete the data return operation, latency is still imposed by the length of the data return path, which extends from the lowest levels of memory hierarchy, to main memory, and back to the lowest memory levels. What is needed, therefore, is a system that minimizes the time required to return data to a requesting processor coupled to the hierarchical memory system by shortening the data return path.

Objects:

The primary object of the invention is to provide an improved shared memory system for a multiprocessor data processing system;

A further object is to provide a hierarchical, directory-based shared memory system having improved response times;

A yet further object is to provide a system for use with a hierarchical memory that transfers data up the hierarchical memory structure in anticipation of receipt of a request to provide the data to the highest level in the memory hierarchy;

Another object is to provide a system that allows modified data residing in first and second-level cache memories to be provided to an associated third-level cache memory in anticipation of the third-level cache memory receiving a request to transfer the data to a main memory;

A yet further object is to provide a system that generates speculative return requests requesting the transfer of data between first and second storage devices included within a hierarchical memory system so that an anticipated fetch operation for the data can be completed more quickly;

A still further object is to provide a hierarchical memory system that allows speculative return requests that are pending to a cache memory to be discarded after the main memory issues a request for the data that is associated with the speculative return request;

Another object is to allow a cache memory to probe one or more associated cache memories to determine the presence of updated data in anticipation of receiving a request for the data;

A still further object is to allow a first cache memory to provide requests to one or more associated cache memories requesting invalidation of predetermined data that may potentially reside within the associated cache memories in preparation for possible receipt by the first cache memory of a request for that predetermined data;

Another object is to allow a first memory to provide requests to one or more associated memories requesting that

a shared copy of data potentially residing within one or more of the associated memories be provided to the first memory in preparation for possible receipt by that first memory of a request for a shared data copy;

Yet another object is to allow a first cache memory to provide requests to one or more associated cache memories requesting that an exclusive copy of data that may potentially reside within the associated cache memories be provided to the first cache memory in preparation for possible receipt by the first cache memory of a request for an exclusive data copy; and

Still another object is to provide a system that allows predetermined fetch requests issued within a data processing system to generate requests to transfer the requested data between various memory resources even before it is known where the latest copy of the data resides.

SUMMARY OF THE INVENTION

The objectives of the present invention are achieved in a speculative return system that generates requests to transfer data between one or more levels within a hierarchical memory structure in anticipation of receiving a request for the data. The hierarchical memory structure includes a main memory coupled to multiple first storage devices, each of which stores data signals retrieved from the main memory. Ones of the first storage devices are further respectively coupled to second storage devices, each of which stores data signals retrieved from the respectively coupled first storage devices. In the preferred embodiment, the first and second storage devices are cache memories, and the main memory is a directory-based memory that includes a directory to indicate which of the other memories is storing a copy of addressable portions of the memory data.

According to the coherency scheme of the hierarchical memory structure, each of the first storage devices is capable of generating a fetch request to the main memory to obtain a copy of requested ones of the data signals. In some instances, the main memory does not store the latest copy of the requested data signals, as will be indicated by corresponding status signals stored in the directory memory. When this occurs, the main memory issues a return request to cause a target one of the first storage devices to return the latest copy of the requested data signals to the main memory so these signals can be forwarded to the requesting storage device. In some cases, however, the target one of the first storage devices, has, in turn, provided the requested data signals to one or more of the respectively coupled second storage devices. Additional storage devices may be further coupled to these second storage devices for storing data copies. Thus, the data signals must be transferred up the hierarchical memory structure, from the storage devices at the lowest level in the memory hierarchy to the target storage device, and finally to the main memory. This imposes latency.

The speculative return system of the current invention decreases the time required for the main memory to retrieve data signals stored in a lower level in the hierarchical memory system. The speculative return system includes at least one speculative return generation logic circuit coupled to at least two of the first storage devices. The speculative return generation logic circuit intercepts fetch requests generated by any of the coupled first storage devices. In response thereto, the speculative return generation logic circuit generates a speculative return request to one or more of the other coupled first storage devices. The speculative return request causes these first storage devices to prepare to send any stored, updated copy of the requested data signals to main memory. This includes retrieving any updated copies of the requested data signals that may be stored at a lower level in the hierarchical memory structure, including those copies stored in the respectively coupled second storage devices.

While any stored copies of the requested data signals are being retrieved in response to the speculative return request, the original fetch request is received by the main memory. In response thereto, the main memory may generate a return request to a target one of the first storage devices to return the latest copy of the requested data signals. If the target one of the first storage devices is one of the one or more storage devices that has already executed the speculative return request, the requested data signals are already resident in the target storage device upon receipt of the return request. These data signals may therefore be provided immediately by the target storage device to the main memory so they can be forwarded to the requesting storage device. This decreases memory latency.

In the current hierarchical memory system, various types of fetch requests may be generated to the main memory. According to one aspect of the speculative return generation system, a speculative return request is generated only in response to the receipt of predetermined types of fetch requests. For example, in the preferred embodiment, some fetch requests are associated with the retrieval of an exclusive data copy, whereas other fetch requests initiate the retrieval of a read-only data copy. Still other types of fetches are conditional fetches that trigger the execution of a prediction algorithm to determine whether an exclusive, versus a read-only copy, will be retrieved. The current speculative return generation system generates speculative return requests for exclusive-copy fetches and some conditional fetches. This design choice is made to minimize the unnecessary transfer of data signals within the hierarchical memory when it is likely that the read-only, shared data copy is already available from the main memory.

According to another aspect of the invention, several types of speculative return requests may be generated depending on the type of fetch request that is issued. In the preferred embodiment, a fetch request that is requesting an exclusive data copy initiates a predetermined type of speculative return request that purges any stored data copy from the lower levels in the memory. Alternatively, a fetch request requesting a shared, read-only data copy initiates a speculative return request that allows lower memory levels to retain a shared, read-only data copy while returning a read-only copy to a respective one of the first storage devices.

The current speculative return system includes logic to temporarily store speculative return requests, if necessary, prior to providing those requests to a respectively-coupled one of the first storage devices for processing in an order determined by a predetermined priority scheme. The speculative return generation system is further coupled to receive from the main memory all return requests that are generated to any of the respectively-coupled ones of the first storage devices. If a return request is received that was initiated by the same fetch request that initiated a still-pending speculative return request, the speculative return request is discarded. The speculative return request is not needed in this instance since the transfer of data from the lower to the higher levels of the memory is accomplished via execution of the return request itself

In one embodiment of the invention, the first storage devices are each associated with a tag memory. This tag

memory stores status signals descriptive of the data signals stored in the associated first storage device, and in additional ones of the storage devices coupled to the associated first storage device at a lower level of the memory hierarchy. These status signals describe both the location and type of any copies of the data signals residing in these storage structures. Speculative return requests issued to first storage devices initiate the return of data signals from lower levels in the memory hierarchy only if the status signals in the tag memory indicate that a predetermined type of data copy exists for the requested data signals. In the preferred embodiment, this data transfer occurs only if an exclusive, read/write copy of the data signals is resident in the lower memory levels. This design choice is made to optimize memory efficiency.

Still other objects and advantages of the present invention will become readily apparent to those skilled in the art from the following detailed description of the preferred embodiment and the drawings, wherein only the preferred embodiment of the invention is shown, simply by way of illustration of the best mode contemplated for carrying out the invention. As will be realized, the invention is capable of other and different embodiments, and its several details are capable of modifications in various respects, all without departing from the invention. Accordingly, the drawings and description are to be regarded to the extent of applicable law as illustrative in nature and not as restrictive.

BRIEF DESCRIPTION OF THE FIGURES

The present invention will be described with reference to the accompanying drawings.

FIG. **1** is a block diagram of a Symmetrical MultiProcessor (SMP) system platform according to a preferred embodiment of the present invention;

FIG. **2** is a block diagram of a Processing Module;

FIG. **3** is a block diagram of the Sub-Processing Module;

FIG. **4** is a block diagram of the TCM of the preferred embodiment;

FIG. **5** is a block diagram of Command/Function Routing Logic;

FIG. **6** is block diagram of the Third Level Cache;

FIG. **7** is a block diagram illustrating the format of requests as provided by the TCM to the MSU;

FIG. **8** is a block diagram illustrating the format of requests provided by the MSU to the TCM;

FIG. **9** is a table summarizing the types of Speculative Return Functions that are generated by the TCM in response to receiving various ones of the Fetch commands from a Sub-POD; and

FIG. **10** is a block diagram of the Speculative Return Generation Logic.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

System Platform:

FIG. **1** is a block diagram of a Symmetrical Multi-Processor (SMP) System Platform according to a preferred embodiment of the present invention. System Platform **100** includes one or more Memory Storage Units (MSUs) in dashed block **110** individually shown as MSU **110**A, MSU **110**B, MSU **11**C and MSU **101**D, and one or more Processing Modules (PODs) in dashed block **120** individually shown as POD **120**A, POD **120**B, POD **120**C, and POD **120**D. Each unit in MSU **110** is interfaced to all PODs **120**A, **120**B, **120**C, and **120**D via a dedicated, point-to-point connection referred to as an MSU Interface (MI) in dashed block **130**, individually shown as **130**A through **130**S. For example, MI **130**A interfaces POD **120**A to MSU **110**A, MI **130**B interfaces POD **120**A to MSU **110**B, MI **130**C interfaces POD **120**A to MSU **110**C, MI **130**D interfaces POD **120**A to MSU **110**D, and so on.

In one embodiment of the present invention, MI **130** comprises separate bi-directional data and bi-directional address/command interconnections, and further includes unidirectional control lines that control the operation on the data and address/command interconnections (not individually shown). The control lines run at system clock frequency (SYSCLK) while the data bus runs source synchronous at two times the system clock frequency (2×SYSCLK).

Any POD **120** has direct access to data in any MSU **110** via one of MIs **130**. For example, MI **130**A allows POD **120**A direct access to MSU **110**A and MI **130**F allows POD **120**B direct access to MSU **110**B. PODs **120** and MSUs **110** are discussed in further detail below.

System Platform **100** further comprises Input/Output (I/O) Modules in dashed block **140** individually shown as I/O Modules **140**A through **140**H, which provide the interface between various Input/Output devices and one of the PODs **120**. Each I/O Module **140** is connected to one of the PODs across a dedicated point-to-point connection called the MIO Interface in dashed block **150** individually shown as **150**A through **150**H. For example, I/O Module **140**A is connected to POD **120**A via a dedicated point-to-point MIO Interface **150**A. The MIO Interfaces **150** are similar to the MI Interfaces **130**, but in the preferred embodiment have a transfer rate that is approximately half the transfer rate of the MI Interfaces because the I/O Modules **140** are located at a greater distance from the PODs **120** than are the MSUs **110**.

Processing Module (POD):

FIG. **2** is a block diagram of a processing module (POD) according to one embodiment of the present invention. POD **120**A is shown, but each of the PODs **120**A through **120**D have a similar configuration. POD **120**A includes two Sub-Processing Modules (Sub-PODs) **210**A and **210**B. Each of the Sub-PODs **210**A and **210**B are interconnected to a Crossbar Module (TCM) **220** through dedicated point-to-point Sub-POD Interfaces **230**A and **230**B, respectively, that are similar to the MI interconnections **130**. TCM **220** further interconnects to one or more I/O Modules **140** via the respective point-to-point MIO Interfaces **150**. TCM **220** both buffers data and functions as a switch between the Sub-POD Interfaces **230**A and **230**B, the MIO Interfaces **150**A and **150**B, and the MI Interfaces **130**A through **130**D. When an I/O Module **140** or a Sub-POD **210** is interconnected to one of the MSUs via the TCM **220**, the MSU connection is determined by the address provided by the I/O Module or the Sub-POD, respectively. In general, the TCM maps one-fourth of the memory address space to each of the MSUs **110**A–**110**D. According to one embodiment of the current system platform, the TCM **220** can further be configured to perform address interleaving functions to the various MSUs. The TCM may also be utilized to perform address translation functions that are necessary for ensuring that each processor (not shown in FIG. **2**) within each of the Sub-PODs **210** and each I/O Module **140** views memory as existing within a contiguous address space as is required by certain off-the-shelf operating systems.

In one embodiment of the present invention, I/O Modules **140** are external to Sub-POD **210** as shown in FIG. **2**. This embodiment allows System Platform **100** to be configured based on the number of I/O devices used in a particular application. In another embodiment of the present invention,